Final Report

EGR 190 Intro to Machine Learning

Vineet Alaparthi, Bianca Saputra, and Ryan Weeratunga

6 December 2019

We have adhered to the Duke Community Standard.

Motivation

Strokes are the third leading cause of death in the United States, yet 73 percent of Americans are unaware of stroke symptoms. Currently, the best way for bystanders to detect when someone is suffering from a stroke is knowing the F.A.S.T. acronym, which involves looking for symptoms of facial droop, arm weakness, and speech difficulty (Ronald Reagan UCLA Medical Center, 2016). The goal of our project is to utilize machine learning to detect whether someone is experiencing facial droop, which is the first indication that someone is having a stroke. Currently, there are no applications available to detect when someone is experiencing a stroke and people may not be aware of the signs they need to look for. By doing this project, people will be able to check if they or others are having a stroke and get help quickly. This will help save lives since people will know if they need immediate attention.

Data

To gather our data, we scraped multiple websites to get our labeled images.For images that showed signs of facial droop, we used images from the Facial Analysis Institute's database of patients with Bell's Palsy. We are using this database because the type of partial facial paralysis that Bell's Palsy causes can be used as a facial proxy for paralysis caused by strokes. For images that showed no sign of facial paralysis, we scraped images from faculty directories because these images were frontal headshots that closely resembled the format of the images with facial droop. Due to the fact that our dataset was relatively small, we increased the size by performing rotations and reflections on the images in keras.

We then preprocessed the data by labeling the photographs showing no signs of facial paralysis as false and the photographs that had facial paralysis were labeled as true. A Python script was written to label the images and assign a specific number to each picture. Using the keras shuffle method, we then split our images into train, validation, and test sets with each set getting 75%, 12.5%, and 12.5% of the images respectively. Additionally, each set had 50% true and 50% false to ensure our model was not biased toward one classification. We shuffled the batches for the training and validation set to ensure that each batch had both true and false labeled data. Once we split our data, we converted the image data into n-dimensional pixel arrays. For our project we had two classes for true and false images. Images that were named true had a label of [1,0] while false images had a label of [0,1]. Next, we reshaped each of the image data into a numpy array of size 128 by 128 and converted all of the images into grayscale images.

Model Architecture

To address our objective, we decided to use our machine learning model as an image classifier that could classify our images as 'TRUE' or 'FALSE' with a true label corresponding to possible facial paralysis and a false label corresponding to a healthy face. Our model is similar to AlexNet, a convolutional neural network designed by Alex Krizhevsky, and was modelled off a tutorial written by a data scientist at Francium Tech (Prakash, A). Our network is five layers deep (see Figure 1). The first three layers are convolutional layers which are each followed by max-pooling layers. The last two layers are fully connected layers, which are called dense layers in keras. Once the images pass through the first three convolution layers, the images were flattened once more in order to pass through the dense layers. Each layer uses a ReLU activation function, with the exception of the last layer which uses a Softmax activation function. For each layer, the filter count increased because the initial layer represented the high level features and the deeper layers represented more detailed features. Before each of the last two layers, we have a dropout rate of 25% and 50% respectively to combat overfitting. This dropout rate was done to prevent overfitting from occurring. The first dense layer had 512 neurons and the second dense layer had 2 neurons, which coincides with our number of classes. This last layer used the softmax function to calculate the probability of each target class for each image. In our model, the probability of true and the probability of false will always add up to 1. For an image, if the probability of true is greater than false, then the model predicts the image is true and shows signs of facial paralysis. If the probability of false is greater than the probability of true, then the model predicts the image is false. For example, if the model calculated a predicted value of [0.8, 0.2] for an image, then the model would label this specific image as 'TRUE' since it had a higher percentage. The Adam optimizer was then used as the loss function and we set the learning rate to 0.001. Our model was trained for 50 epochs and our model had a validation accuracy of 75% and a test accuracy of 85%.

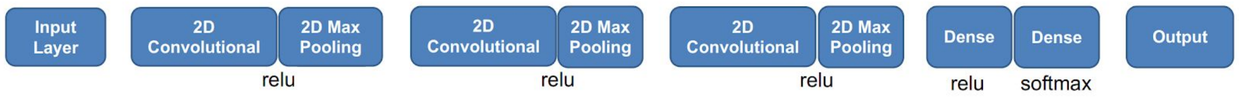

Figure 1: This illustrates our model architecture.

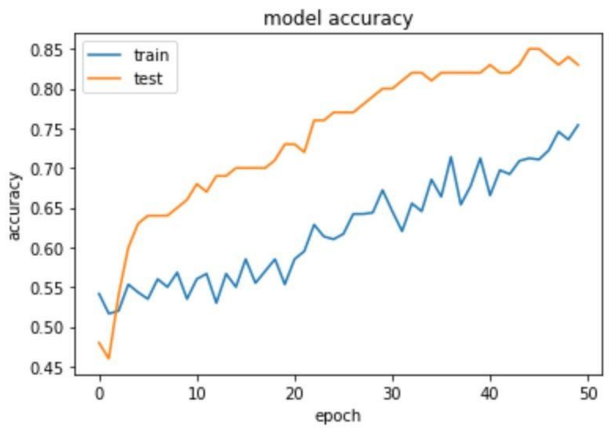

Figure 2: Graph of model accuracy with respect to number of epochs

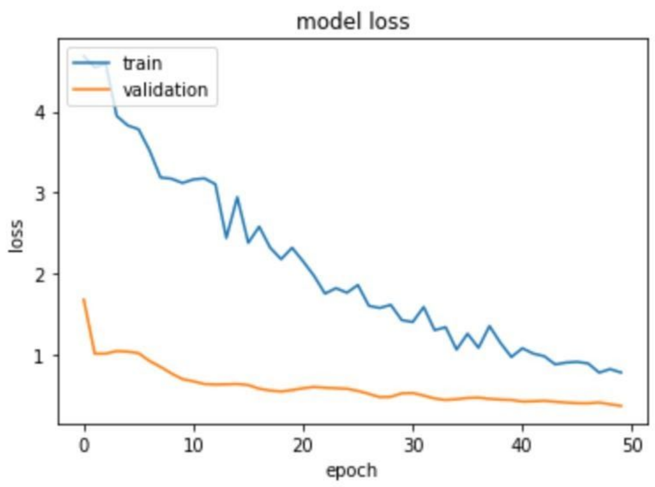

Figure 3: Graph of model loss with respect to number of epochs

Conclusion

When testing our model against the validation set, images were labeled correctly with an accuracy of 75%; however, when the model was tested against the test set, images were labeled correctly with an accuracy of 85%. This accuracy indicates that our model was satisfactory and was close to our goal of 70% accuracy. On the other hand, this accuracy may have been artificially inflated because our dataset was relatively small since our total dataset only had 600 images. It is unorthodox that the test accuracy was larger than the validation accuracy because typically the model overfits the training and validation data, resulting in the model not performing as well on the test set and on real-life applications. However, due to our small dataset, our standard deviation for accuracy is potentially large and likely explains this result. When first running the model, we found that our image classification model seemed to be overfitting the training and validation data so we increased the dropout rate in an attempt to minimize this error.

Future Work

Even though our model has a relatively high accuracy and is better than pure chance, it is not ready to be implemented into practice. An extension of this project could be to utilize transfer learning and reweight the features of the pre-trained model on the top layers of the Convolution Neural Network (CNN). A pre-trained model such as MobileNet would improve the accuracy of our model because it already have specified parameters that have proven success. Even though we changed our parameters several times to get better accuracy, it is likely that a pre-trained model will perform better than ours.

References

Bosler, F. (2019, September 27) Image Scraping with Python.

Facial Paralysis: Bell's Palsy: Los Angeles, CA.(n.d)

Facial Paralysis Photo Gallery: Facial Paralysis Institute. (n.d.)

Facial Paralysis Surgery: Johns Hopkins Medicine. (2019, February 12).

Prakash, A. (2018, June 4). Build your own Image classifier with Tensorflow and Keras.